REMARKS

In the Official Action mailed on **11 March 2007**, the Examiner reviewed claims 1, 3-5, 9-10, 12, 15, 17-19, 23-24, 26, 29, 31-33, 37-38, and 40. Examiner rejected claims 1, 4, 6-9, 11, 13-14, 15, 18, 20-23, 25, 27-28, 29, 32, 34-37, 39, and 41-42 under 35 U.S.C. § 103(a) based on McGregor (*"Practical C++*, published by Que on 11 August 99, hereinafter "McGregor"), and Guthrie II (U.S. Patent No. 7,225,210, hereinafter "Guthrie"). Examiner rejected claims 3, 17, and 31 under 35 U.S.C. § 103(a) based on McGregor, Guthrie and Official Notice.

Rejections under 35 U.S.C. §103(a)

Independent claims 1, 15, and 29 were rejected based on McGregor and Guthrie. Applicant respectfully points out that both McGregor and Guthrie nowhere disclose that the process of copying the snapshot of an existing node to a new node involves **examining the next pointer of the existing node to determine if the existing node has been deleted**.

More specifically, while copying the entire contents (i.e., the snapshot) of the existing node into the new node, the embodiments of the present invention additionally examine the next pointer in the existing node to determine if existing node has been deleted. If not, that is, if the existing node is not deleted and therefore is "current," the system determines that the snapshot has been successfully copied. On the other hand, if it is found that the existing node has been deleted, the system determines that the snapshot is invalid because the existing node has been previously deleted and therefore the contents in the existing node are invalid. In this case, the system can then follow the next pointer of the existing node in an attempt to find an updated version of existing node or take other remedial actions (see paragraph [0038] and [0039] of the instant application).

Applicant respectfully points out that McGregor describes a list of standard linked list operations for the doubly linked list, such as creating a list, adding, inserting and deleting a node. Examiner asserts that McGregor discloses "examining the next pointer of the existing node to determine if the existing node has been deleted" based on the "CIntList::Find( ) function" in page 8. Applicant respectfully points out that **the CIntList::Find( ) function in McGregor is used to find an item (i.e., a node) in the linked list by traversing the list**. Hence, CIntList::Find( ) function merely performs a traversing operation (see McGregor, "Find an item," in pages 8-9). The description of CIntList::Find( ) function in McGregor **neither involves examining a next pointer for a node nor determining if a traversed node has been deleted**. Because the CIntList::Find( ) function is not used for copying a node, there is no motivation for examining a next pointer to determine if a traversed node is deleted.

We now turn to Guthrie, which describes a snapshot system configured to create and make available multiple snapshots representing different states of the data at various times (see Guthrie. Col. 7, lines 15-20). More specifically, Guthrie describes various techniques for making snapshots of existing nodes by copying the contents of the existing nodes into new nodes and then modifying pointers to add the new nodes into the linked list. However, the snapshots copying process described by Guthrie does not suggest or imply **examining the next pointer of an existing node to determine if the existing node has been deleted.**

In summary, there is nothing within McGregor and Guthrie, either separately or in concert, which suggests that the process of copying the contents of an existing node involves **examining the next pointer of the existing node to determine if the existing node has been deleted**.

Accordingly, Applicant has amended independent claims 1, 15, and 29 to clarify that embodiments of the present invention provide a detection mechanism while copying the snapshot of an existing node by examining the next pointer of

the existing node to determine if the existing node has been deleted. These amendments find support in paragraph [0038] and [0039] of the instant application. No new matter has been added.

Hence, Applicant respectfully submits that independent claims 1, 15, and 29 as presently amended are in condition for allowance. Applicant also submits that claims 3-14, which depend upon claim 1, claims 17-28, which depend upon claim 15, and claims 31-42, which depend upon claim 29, are for the same reasons in condition for allowance and for reasons of the unique combinations recited in such claims.

CONCLUSION

It is submitted that the application is presently in form for allowance. Such action is respectfully requested.

Respectfully submitted,

By /Anthony Jones/
Anthony Jones
Registration No. 59,521

Date: 11 June 2008

Anthony Jones
Park, Vaughan & Fleming LLP
2820 Fifth Street
Davis, CA 95618-7759
Tel: (530) 759-1666
Fax: (530) 759-1665
Email: tony@parklegal.com